รีวิว

สรรค์ เสกขุนทด

ก้าวกระโดดอีกก้าวกับ Microsoft C/C++ 7.0

พัฒนาตัวเองเพื่อขยับตัวให้สูงขึ้นกับภาษาคอมพิวเตอร์ที่กำลังมาแรงในยุคปัจจุบัน

ไมโครซอฟต์ขยับตัวอีกครั้งสำหรับก้าวกระโดดอีกก้าวในวงการซอฟต์แวร์เกี่ยวกับตัวแปรภาษาและเครื่องมือพัฒนาโปรแกรม คราวนี้ไมโครซอฟต์มากับภาษาที่กำลังได้รับความนิยมในปัจจุบัน คือ C++ พร้อมกับซอฟต์แวร์สนับสนุนอีกเป็นจำนวนมาก หลังจากที่ปล่อยให้บอร์แลนด์ล้ำหน้าไปก่อน

Microsoft C/C++ 7.0 หรือเรียกย่อ ๆ ว่า C7.0 เป็นซอฟต์แวร์เวอร์ชันล่าสุดต่อจาก Microsoft C 6.0 ประกอบด้วยคุณสมบัติที่สำคัญ คือ สนับสนุน C++ (AT&T C++ 2.1), มี precompiled header, เทคนิคการออปติไมซ์ p-code, ตัวจัดการโอเวอร์เลย์และคลาสไลบรารีสำหรับไมโครซอฟต์วินโดวส์และดอส

- เป็นที่น่ายินดีสำหรับผู้พัฒนาโปรแกรมบนไมโครซอฟต์วินโดวส์ คือ C7.0 มีเครื่องมือพัฒนาโปรแกรมบนไมโครซอฟต์วินโดวส์อย่างครบครันโดยไม่ต้องไปพึ่ง SDK (Software Development Kit) อีกเลย ที่มากับ C7.0 ก็ได้แก่เฮดเดอร์ไฟล์ต่าง ๆ, ไลบรารี, ไดอะล็อกและรีซอร์สเอดิเตอร์และเคอร์นัลสำหรับการดีบัก

386 หรือ 486 พร้อมด้วยแรม 4 เมกะไบต์ ฮาร์ดดิสค์ว่างอย่างน้อย 10 เมกะไบต์ ดอส 3.3 ขึ้นไป จอ CGA, EGA หรือ VGA (ควรจะเป็น EGA ขึ้นไป) ที่กล่าวมานั้นเป็นฮาร์ดแวร์ที่จำเป็นสำหรับ C7.0

มีอะไรใหม่ใน C7.0

ความสามารถใหม่ของ C7.0 ที่ควรพูดถึงมากที่สุดคือ การสนับสนุนภาษา C++ ภาษา C++ นั้นเป็น super set ของภาษาซี หมายความว่าโปรแกรมภาษาซีสามารถคอมพายล์ด้วยคอมพายเลอร์ C++ ได้โดยไม่ต้องแก้ไขโปรแกรมเลย ถ้าหากเปลี่ยนนามสกุลจาก .C เป็น .CPP แล้ว

คอมพายล์ด้วย C7.0 C7.0 จะเห็นเป็นโปรแกรมภาษา C++ ทันที และจะใช้กฎเกณฑ์ของภาษา C++ กับโปรแกรมนั้น เช่น เข้มงวดเรื่อง type, สามารถโอเวอร์โหลดฟังก์ชันได้, ประกาศตัวแปรตรงไหนก็ได้ (ของภาษาซีต้องประกาศต้นบล็อกหรือนอกฟังก์ชัน) เป็นต้น

สำหรับโปรแกรมเมอร์ C++ C7.0 ได้เตรียมคลาสไลบรารีสำหรับพัฒนาทั้งบนดอสและไมโครซอฟต์วินโดวส์ ในส่วนของดอสนั้น C7.0 ได้เตรียมคาสไลบรารี ซึ่งเรียกว่า MFC (Microsoft Foundation Class) ไว้กว่า 60 คลาส เช่น คลาสวินโดว์, ไดอะล็อก, เมสเสจ, คอนโทรล และคลาส GDI ต่าง ๆ นอกจากนี้ยังมีคลาสสำหรับ MDI (Multiple Document Interface) และ OLE (Object Linking and Embadding-มีเฉพาะวินโดวส์ 3.1) ด้วย ที่สำคัญก็คือ คลาสพื้นฐาน เช่น สตริง, ไฟล์, ลิสต์ ฯลฯ สามารถใช้ทั้งบนดอสและวินโดวส์ได้

ออปชัน, ตัวแปรและคีย์เวิร์ดที่เพิ่มเติมใน C7.0

C7.0 มีออปชัน, ตัวแปร และคีย์เวิร์ดเพิ่มขึ้นมากมาย รูปที่ 1 แสดงออปชันที่เพิ่มขึ้นมาใน C7.0 มีคีย์เวิร์ดใหม่คำหนึ่ง คือ __inline ใช้สำหรับฟังก์ชันเพื่อให้คอมพายเลอร์ทำ "inline" คือ "แปะ" ฟังก์ชันลงไปตรงตำแหน่งที่มีการเรียกใช้ฟังก์ชันนั้น โดยไม่มีการ call ฟังก์ชันจริง ๆ นอกจากนี้ก็มีฟังก์ชัน __commit และ __doscommit ที่ใช้ flush ไฟล์บัฟเฟอร์โดยตรง ส่วน Quick C นั้นสามารถใช้ฟังก์ชันอย่าง printf ในโปรแกรมบนวินโดวส์ได้เหมือนกับเขียนโปรแกรมบนดอส (ความสามารถนี้มีใน Borland C++ 3.0 ด้วย)

สำหรับโปรแกรมบนดอส C7.0 มีฟังก์ชันจัดการเวอร์ชวลเมมโมรี มีตัวแปร __cpumode เพื่อบอกว่าโปรแกรมกำลังรันอยู่ในโหมดอะไร (real หรือ protect) นอกจากนี้ไมโครซอฟต์ได้ปรับปรุงคำเตือนและรายงาน error ต่าง ๆ ได้ดีขึ้นจากเวอร์ชันเดิม รวมทั้งทำการตัดและเปลี่ยนแปลงความสามารถบางอย่างใน C6.0

รูปที่ 1 แสดงออปชันที่เพิ่มขึ้นมาใน C7.0

Figure 1 *New Command-line Options*

Precompiled Header Control	
/Fp*filename*	Precompiled header filename
/Yc*filename*	Create precompiled header
/Yd	Include debug information in precompiled header
/Yu	Use precompiled header file
Code Generation	
/G3	Use 80386 instructions
/G4	Use 80486 instructions
/Ob*number*	Control inline expansion
/Oo[-]	Toggle post-code-generation optimizing
/Gq	Specify real mode Windows (compatible with /Gw)
/Gx	Assume all near data
/Gy	Enable function-level linking
P-code Generation and Control	
/Oq	Turn on p-code optimization
/Ov[-]	Sort local variables in p-code
/Of[-]	Toggle p-code quoting
/Gn	Remove p-code native entry points
/Gp*number*	Specify maximum number of entry tables
/NQpc*odeseg*	Name temporary segment for p-code
Other Optimization Options	
/GA	Optimize entry/exit code for protected-mode Windows applications
/GD	Optimize entry/exit code for protected-mode Windows DLLs
/GE*string*	Optimize entry/exit code for protected-mode Windows DLLs
Linker Control	
/Ld	Control library selection for DLLs
/Lw	Control library selection for applications
/Lu	Link without C run-time startup
Miscellaneous Options	
/Bm*memavail*	Sets amount of memory available to compiler
/F	Fast compile
/Zf	Accept __far keyword
/Zn	Turn off SBRPACK utility
/NV	Name temporary segment for far virtual tables
/Tp	Specify C++ source file

C++ ใน C7.0

C7.0 นั้นครอบคลุมถึง C++ ของ AT&T 2.1 และบางส่วนของ C++ AT&T 3.0 โดยเฉพาะเรื่อง Template ซึ่งยังคงเป็นที่ถกเถียงกันระหว่าง AT&T กับคณะกรรมการ ANSI C++ ว่าจะจัดการอย่างไรกับเรื่อง Template นี้

บอร์แลนด์และไมโครซอฟต์หาทางออกในเรื่อง Template สำหรับผลิตภัณฑ์ของตนในแนวทางต่างกัน โดยบอร์แลนด์จะจัดการ Template ให้เองโดยคอมพายเลอร์เป็นตัวจัดการ สำหรับไมโครซอฟต์จัดการโดยผู้ใช้จะต้องไปเรียกโปรแกรมจัดการ Template เสียก่อนที่จะคอมพายล์

สำหรับผู้ที่ไม่คุ้นเคย C++ อาจจะไม่รู้จัก Template ก็ขออธิบายพอเป็นสังเขปปกติในการออกแบบคลาสนั้นจะพยายามให้คลาสนั้นจัดการกับดาต้าชนิดอะไรก็ได้ ซึ่งจะเรียกว่า generic class และตัว Template นี้เองที่จะเป็นตัวช่วยจัดการในเรื่องดังกล่าว

ฟังก์ชันและ auto-lining

Inlining คือ ความสามารถในการนำตัวฟังก์ชันที่ถูกเรียกไป "แปะ" ลงไปยังตำแหน่งที่เรียกฟังก์ชันโดยไม่มีการ call ฟังก์ชันจริง ๆ (คล้าย ๆ แมคโคร) ประโยชน์ของ inline คือเพิ่มความเร็วของโปรแกรม แต่จะทำให้โปรแกรมมีขนาดโตขึ้นด้วย

ฟังก์ชันใด ๆ ของ C++ สามารถประกาศให้เป็น inline ได้ แต่ในแง่ของการอินพลีเมนต์แล้วค่อนข้างยากพอสมควร Borland C++ 2.0 และ 3.0 ไม่อนุญาตให้ inline ฟังก์ชันที่มี switch, while และ for ในตัวฟังก์ชัน สำหรับ C7.0 นั้นเหนือชั้นกว่าบอร์แลนด์ในเรื่องนี้ โดยผู้ใช้สามารถ inline ฟังก์ชันใด ๆ ก็ได้ ออปชันสำหรับ inline ได้แก่ /Ob0, /Ob1 และ /Ob2 (ดูรายละเอียดจากคู่มือ C7.0)

CodeView และ PWB

codeView 4.0 เป็นโปรแกรมดีบักเกอร์สำหรับ C7.0 ยูสเซอร์อินเตอร์เฟสยังคงเป็น text.mode คุณสมบัติดี ๆ ก็เช่น คำนวณนิพจน์ของ C++ ได้ ตั้งเบรกพอยน์ และ trace เข้าไปยัง constructor และ destructor ได้ trace เข้าไปยังฟังก์ชันแบบ inline และ virtual ได้ นอกจากนี้ก็มี class browsing ไว้แสดงความสัมพันธ์ระหว่างคลาสต่าง ๆ

ส่วน PWB เป็นโปรแกรมจัดการเกี่ยวกับการพัฒนาซอฟต์แวร์ ประกอบด้วยส่วนที่เป็นเอดิเตอร์, help ในการเขียนโปรแกรม รวมทั้งคอมพายล์และลิงค์เช่นเดียวกับ codeView 4.0, PWB มียูสเซอร์อินเตอร์เฟสแบบ text-mode

Precompile & Headers

Precompile & header เป็นแนวคิดเพื่อต้องการลดเวลาในการคอมพายล์โปรแกรม หลักการคือ ทุกครั้งที่คอมพายล์ไฟล์ .C ไฟล์ .H ก็จะถูกคอมพายล์ใหม่เช่นเดียวกัน ดังนั้นถ้าคอมพายล์ .H เก็บไว้ เมื่อคอมพายล์ .C อีกครั้งก็ไม่จำเป็นต้องคอมพายล์ .H อีก เป็นการลดเวลาในการพัฒนาโปรแกรม

precompiled header ไม่ใช่แนวคิดใหม่ บอร์แลนด์เป็นเจ้าของความคิดนี้ สำหรับ C7.0 ไมโครซอฟต์ก็นำมาประยุกต์และทำได้ดีกว่าของบอร์แลนด์ด้วย (อาจเป็นเพราะว่าโปรแกรมได้ออกมาทีหลัง) ผู้ใช้สามารถ precompile ไฟล์ .C ได้ด้วย โดยต้องระบุจุดหยุดการ pricompile ในไฟล์ .C

ปกติแล้ว เรามักจะเปลี่ยนแปลงแก้ไขไฟล์ .C มากครั้งกว่าไฟล์ .H เช่น กรณีที่ไฟล์ .H ใหญ่มาก (เช่น windows.H) ถ้าต้องคอมพายล์ .H ทุกครั้งก็ยิ่งเสียเวลามาก ดังนั้น precompile header จะช่วยลดเวลาในการพัฒนาโปรแกรมได้เป็นอย่างมาก

MFC

นอกเหนือไปจากการสนับสนุน C++ แล้ว สิ่งที่เป็นจุดเด่นของ C7.0 อีกอย่างหนึ่งคือ MFC (Microsoft Foundation Class) MFC เป็นคลาสไลบรารีสำหรับใช้พัฒนาโปรแกรมทั้งบนดอสและวินโดวส์ ประกอบด้วยคลาสกว่า 60 คลาส สำหรับวินโดวส์คลาสของ MFC มีความคอมแพตทิเบิลทั้งวินโดวส์ 3.0 และ 3.1 รวมทั้งคลาสสำหรับ MDI และ OLE

MFC ถ้าจะเปรียบแล้วก็ต้องเปรียบกับ OWL (ObjectWindows Library) ของบอร์แลนด์ซึ่งเป็นคลาสไลบรารี เช่นเดียวกัน แต่ในแง่ความสามารถและลักษณะการใช้แล้ว MFC ค่อนข้างใช้ง่ายกว่าและให้โค้ดที่ไม่ใหญ่จนเกินไปนัก เมื่อเทียบกับโปรแกรมเดียวกันที่เขียนด้วยภาษาซี ถ้าเขียนโปรแกรมด้วย MFC โปรแกรมเมอร์สามารถพัฒนาโปรแกรมได้เร็วขึ้นกว่าเดิม เมื่อเทียบกับการเขียนด้วยภาษาซี โปรแกรมเมอร์เพียงแค่ดีไรว์ ส่วนที่ต้องการเปลี่ยนแปลงเท่านั้น ไม่จำเป็นต้องเขียนใหม่หมด ด้วยวิธีนี้เองทำให้โปรแกรมเมอร์ลดเวลาในการพัฒนาโปรแกรมลงได้

สำหรับการเขียนโปรแกรมบนวินโดวส์โดยใช้ MFC เกี่ยวกับเรื่องเมสเสจนั้น C7.0 ก็ได้เตรียมการไว้ให้ใช้ได้ง่าย ๆ โดยโปรแกรมเมอร์เพียงแค่ระบุไปว่าจะดักเมสเสจใดมาใช้งานด้วยวิธีการที่เรียกว่า message mapping ซึ่งจะเป็นตัวบอกว่าเมสเสจใดที่ถูกดักโดยฟังก์ชันอะไร

MFC ได้เตรียมคลาสทั่ว ๆไป เช่น สตริง ไฟล์ ลิสต์ เป็นต้น ไว้ให้โปรแกรมเมอร์ใช้อย่างพร้อมมูล (ทั้ง

รูปที่ 2 แสดง Source โปรแกรมของ MFCHELLO

Figure 4 *MFCHELLO*

MAKEFILE

```
# Makefile : Builds the mfchello application

CPPFLAGS= /AS /W3 /GA /GEs /G2
LINKFLAGS=/NOD

!if "$(DEBUG)"=="1"
CPPFLAGS=/D_DEBUG $(CPPFLAGS) /Od /Zi /f
LINKFLAGS=$(LINKFLAGS) /COD
LIBS=safxcwd libw slibcew
!else
CPPFLAGS=$(CPPFLAGS) /Oselg /Gs
LINKFLAGS=$(LINKFLAGS) /FAR /PACKC
LIBS=safxcw libw slibcew
!endif

mfchello.exe:     mfchello.obj mfchello.def mfchello.res
        link $(LINKFLAGS) mfchello, mfchello, NUL, $(LIBS),mfchello.def;
        rc /t mfchello.res


mfchello.res:     mfchello.h mfchello.ico mfchello.dlg

mfchello.obj:     mfchello.h mfchello.cpp

clean:
        -erase mfchello.exe
        -erase mfchello.res
        -erase mfchello.obj
```

MFCHELLO.DEF

```
NAME          MFCHello
DESCRIPTION   'MFCHello Windows App - 1992, Richard Hale Shaw'

EXETYPE       WINDOWS
STUB          'WINSTUB.EXE'

CODE          PRELOAD MOVEABLE DISCARDABLE
DATA          PRELOAD MOVEABLE MULTIPLE

HEAPSIZE      1024
STACKSIZE     4096
```

MFCHELLO.RC

```
// mfchello.rc RHS

#include<windows.h>
#include<afxres.h>
#include"mfchello.h"

MainMenu MENU
    {
    POPUP       "&Help"
        {
        MENUITEM "&About Hello...\tF1", IDM_ABOUT
        }
    }

MainAccelTable ACCELERATORS
    {
    VK_F1,      IDM_ABOUT,  VIRTKEY
    }

ICON 1 ICON mfchello.ico

rcinclude mfchello.dlg
```

MFCHELLO.H

```
// mfchello.h RHS

#define IDM_ABOUT   100
```

รูปที่ 2 (ต่อ)

MFCHELLO.CPP

```
// mfchello.cpp RHS

#include<AFXWIN.H>
#include"mfchello.h"

#define IDM_ABOUT   100

class HelloWindow : public CFrameWnd    // main window derived from CFrameWnd
    {
public:
    HelloWindow();              // class constructor
    afx_msg void OnPaint();     // declare message handling functions
    afx_msg void OnAbout();
    DECLARE_MESSAGE_MAP()       // declare message map
    };

HelloWindow::HelloWindow()      // constructor for HelloWindow
    {
    LoadAccelTable("MainAccelTable");
    Create(NULL, "MFCHello", WS_OVERLAPPEDWINDOW, rectDefault,
           NULL, "MainMenu");
    }

void HelloWindow::OnPaint()     // WM_PAINT handling
    {
    CPaintDC dc(this);
    dc.SetTextAlign(TA_BASELINE | TA_CENTER);
    dc.SetBkMode(TRANSPARENT);

    CRect rect;
    GetClientRect(rect);

    CString s = "Hello Windows, from MFC!";
    dc.TextOut((rect.right/2), (rect.bottom/2), s, s.GetLength());
    }

void HelloWindow::OnAbout()     // WM_COMMAND with wParam==IDM_ABOUT handling
    {
    CModalDialog about("AboutBox", this);
    about.DoModal();
    }

BEGIN_MESSAGE_MAP(HelloWindow, CFrameWnd) // message map definition
    ON_WM_PAINT()
    ON_COMMAND(IDM_ABOUT, OnAbout)
END_MESSAGE_MAP()

class MFCHello : public CWinApp // main application class derived from CWinApp
    {
public:
    BOOL InitInstance();        // declare InitInstance member function
    };

BOOL MFCHello::InitInstance()   // define InitInstance member function
    {
    m_pMainWnd = new HelloWindow();
    m_pMainWnd->ShowWindow(m_nCmdShow);
    m_pMainWnd->UpdateWindow();
    return TRUE;
    }

MFCHello myApp;                 // instantiate the application class,
                                // and start the program
```

ดอสและวินโดวส์) นอกจากนี้ยังมี TEMPLDEF ไว้จัดการเรื่อง Template ด้วย

คลาสทุกคลาสของ MFC จะถูกดีไรว์มาจากคลาสต้นสุดเสมอ เรียกว่า CObject ซึ่งเป็นไปตามมาตรฐานของภาษา C++ ทั่ว ๆ ไป CObject จะทำหน้าที่เกี่ยวกับ type checking และอื่น ๆ ซึ่งจะทำให้ออปเจ็กต์ทุกตัวสามารถตรวจสอบ type ขณะรันได้

โปรแกรม MFCHELLO

สำหรับบทความนี้จะได้แสดงการเขียนโปรแกรมบนวินโดวส์ง่าย ๆ โดยใช้ MFC สำหรับผู้ที่คุ้นเคยกับการเขียนโปรแกรมวินโดวส์จะได้เห็นว่า MFC ช่วยให้เขียนโปรแกรมได้ง่ายขึ้นและโปรแกรมสั้นลง

โปรแกรม MFCHELLO แสดงดังรูปที่ 2 จะสร้างหน้าต่างขึ้นมาประกอบด้วย title bar, system icon, minimize/maximize box และเมนูที่จะไปเรียก AboutBox โปรแกรมนี้จะแสดงข้อความ "Hellow Windows, from MFC" ที่หน้าต่าง

โปรแกรมที่ใช้ MFC ทุกโปรแกรมจะต้องมีแอพพลิเคชันคลาส ซึ่งจะได้จากคลาส CWin_App ของ MFC คลาสนี้ (CWinApp) จัดเตรียม Win Main และเมสเสจลูปไว้ให้เรียบร้อย โปรแกรมเมอร์ไม่ต้องเขียนเอง นอกจากนี้ก็มีคลาส CFrameWnd ซึ่ง MFC ได้จัดเตรียมไว้ให้ทำหน้าที่เกี่ยวกับการสร้างและจัดการหน้าต่าง

เรามาดูโปรแกรมทีละส่วนไฟล์ AFXWIN.H เป็นเฮดเดอร์ไฟล์ที่ประกาศคลาสต่าง ๆ และรวม windows.h เข้าไปด้วย ถัดมาก็คือ การประกาศคลาส HelloWindow ซึ่งดีไรว์มาจากคลาส cFrameWnd คลาส Hellowindow ประกอบด้วย membar function 3 ฟังก์ชัน เป็น constructor 1 ฟังก์ชัน ที่เหลือเป็นฟังก์ชันจัดการเมสเสจ ตามหลังฟังก์ชันทั้งสามเป็นการประกาศ message mapping ซึ่งจะมีรายละเอียดต่อไป

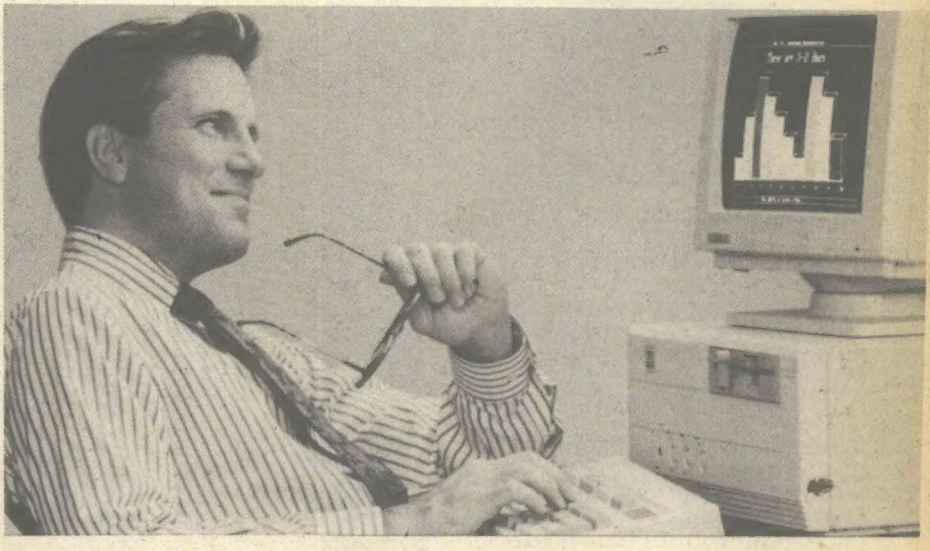
ฟังก์ชัน HelloWindow จะเรียกฟังก์ชัน LoadAccelTable เพื่อทำการโหลด accelerator table หลังจากนั้นจึงทำการสร้างหน้าต่างขึ้นมาโดยการเรียก member function ที่ชื่อ Create (inherit จาก cFrameWnd)

สำหรับฟังก์ชัน OnPaint จะทำหน้าที่แสดงข้อความ "Hellow Windows, from MFC" ที่หน้าต่างเมื่อปรากฏเมสเสจ WM_PAINT เข้ามา ภายในฟังก์ชัน OnPaint จะมีการสร้างออปเจ็กต์คลาส CRect มีการสร้างออปเจ็กต์ของคลาส CString เพื่อเก็บข้อความที่ต้องการแสดง และสุดท้ายที่ OnPaint จะเรียกไปยังฟังก์ชัน Textout ซึ่งเป็น member function ของคลาส CPaintDC เพื่อใช้แสดงข้อความ

ถัดจากฟังก์ชัน OnPaint คือฟังก์ชัน OnAbout ซึ่งจะแสดงไดอะล็อกบ็อกซ์เท่านั้นเอง ภายในฟังก์ชันมีการใช้คลาส CModalDialog เพื่อโชว์ About Box

หลังจากนั้นจะเป็น message mapping จากโปรแกรมจะอธิบายได้ดังนี้ ฟังก์ชัน ON_WM_PAINT จะแม็พเมสเสจ WM_PAINT เข้ากับฟังก์ชัน onPaint เสมอ ดังนั้นจึงประกาศแต่ On_WM_PAINT ส่วนฟังก์ชัน ON_COMMAND จะแม็พกับเมสเสจ WM_COMMAND โดยผู้ใช้จะต้องบอกว่า wParam มีค่าเท่าไร และเป็นฟังก์ชันอะไร (ในที่นี้คือ IDM_ABOUT และ OnAbout ตามลำดับ)

หลัง message mapping เป็นการประกาศคลาส AFCHello ซึ่งจะดีไรว์มาจากคลาส CWinApp ภายในคลาสมี member function อันเดียวคือ InitInstance ซึ่งทำหน้าที่สร้างออปเจ็กต์ของคลาส HellowWindow และสั่งโชว์หน้าต่าง



สุดท้ายคือการประกาศออปเจ็กต์ของคลาส MFC Hello ซึ่ง Constructor ของคลาสนี้จะเรียกฟังก์ชัน Init Instance โดยอัตโนมัติ หลังจากนั้นโปรแกรมจะเข้าเมสเสจลูปต่อไป

จากที่เห็นนั้น โปรแกรมที่สร้างด้วย MFC จะค่อนข้างสั้นไม่เยิ่นเย้อ ส่วนไหนที่ต้องใช้ซ้ำ ๆ ไม่เปลี่ยนแปลง MFC ก็จัดการให้ ซึ่งช่วยลดเวลาการโค้ดไปมาก

P-code

คุณสมบัติดีเยี่ยมอีกประการหนึ่งของ C7.0 คือ การออปติไมซ์โค้ดด้วยวิธีการที่เรียกว่า p-code แอพพลิเคชันหลายตัวของไมโครซอฟต์ที่วางขายในท้องตลาดใช้เทคนิค p-code นี้ ในการพัฒนาโปรแกรมผลที่ตามมา คือ โปรแกรมที่ขนาดเล็กลงโดยความเร็วไม่สูญเสียไปมากนัก

ตามปกติขนาดของโปรแกรมและความเร็วของโปรแกรมมักจะสวนทางกัน ถ้าต้องการให้โปรแกรมเร็วขนาดของโปรแกรมก็จะเพิ่มขึ้นและในทำนองกลับกัน ถ้าต้องการให้โปรแกรมมีขนาดเล็กลง ความเร็วของโปรแกรมก็จะช้าลง

แนวความคิดของ p-code คือ ทำอย่างไรจึงจะลดขนาดโปรแกรมลงไปโดยไม่สูญเสียความเร็ว ไมโครซอฟต์จึงได้คิดค้นเทคนิค p-code นี้ขึ้นมา p-code เป็นภาษาในระดับสูงกว่าภาษาแอสเซมบลี โดย 1 คำสั่งของ p-code ประกอบด้วยคำสั่งภาษาแอสเซมบลีหลายคำสั่ง ภายในโปรแกรมที่ใช้เทคนิค p-code จะมีอินเตอร์เพรตเตอร์ ภาษา p-code เมื่อโปรแกรมรันมาถึงจุดที่ออปติไมซ์ด้วย p-code อินเตอร์เพรตเตอร์ก็จะทำงานแปลคำสั่ง p-code ให้กลายเป็นแมชชีนโค้ดและเอ็กซีคิวต์ ยกตัวอย่างเช่น ภาษาแอสเซมบลี ดังนี้

```
pop CX ; pop 1st operand into CX
pop DI ; pop 2st operand into DI
add DI, CX ; add and store result in DI
push DI ; push result onto stack
```

สามารถแปลงเป็นคำสั่ง p-code ได้ 1 คำสั่งคือ

```
AddW
```

ภาษา p-code เป็นภาษาที่ใช้สแต็กเป็นหลัก หมายความว่า โอเปอร์แรนด์ของ p-code จะถูกดำเนินการบนสแต็ก เช่น อ่านโอเปอร์แรนด์จากสแต็กมาบวกกันแล้วนำไปเก็บไว้บนสแต็ก เป็นต้น

เมื่อไรจึงจะใช้ p-code

เราสามารถเลือกได้ว่าจะให้ส่วนไหนของโปรแกรม (บางฟังก์ชันบางโมดูลหรือทั้งโปรแกรม) ออปติไมซ์ด้วย p-code ซึ่งถือว่าเป็นประโยชน์มาก เพราะในบางส่วนของโปรแกรมที่ต้องการความเร็วสูงก็ไม่ควรใช้ p-code

แอพพลิเคชันที่เหมาะสำหรับใช้ p-code ทั้งโปรแกรมควรจะเป็นโปรแกรมที่ไม่กินเวลาซีพียู และหนักไปทางยูสเซอร์อินเตอร์เฟส เช่น เวิร์ดโปรเซสเซอร์ เอดิเตอร์ เป็นต้น

C7.0 ได้เตรียมเครื่องมือที่วิเคราะห์ว่า ฟังก์ชันไหนของโปรแกรมที่ใช้บ่อยหรือนาน ๆ ใช้ที เครื่องมือนี้มีชื่อว่า Profiler ผลลัพธ์ที่ได้จากการโปรไฟล์โปรแกรมจะเป็นเครื่องบ่งชี้ว่า ฟังก์ชันไหนควรหรือไม่ควรใช้ p-code ในเรื่องนี้มีข้อแนะนำว่า ฟังก์ชันไม่ถูกเรียกใช้บ่อยหรือฟังก์ชันจัดการข้อผิดพลาดหรือฟังก์ชันที่เป็นยูสเซอร์อินเตอร์เฟสควรจะใช้เทคนิค p-code มากที่สุด ส่วนฟังก์ชันที่ไม่ควรใช้ p-code ได้แก่ ฟังก์ชันที่ต้องการความเร็วสูงและฟังก์ชันที่ถูกเรียกใช้บ่อย ๆ

แอพพลิเคชันบางตัวที่มีขนาดเล็กมากก็ไม่ควรใช้ p-code ขนาดของอินเตอร์เพรตเตอร์ประมาณ 9 กิโลไบต์ ไมโครซอฟต์แนะนำว่าโปรแกรมที่จะใช้ p-code ควรมีขนาด 25 กิโลไบต์ขึ้นไป

สำหรับเรื่องดีบักแล้ว codeView 4.0 ที่มาพร้อมกับ C7.0 สามารถดีบักซอร์สของ p-code ได้ สามารถตั้งเบรคพอยน์ในคำสั่ง p-code ได้ นอกจากนี้ codeView 4.0 ยังสามารถแสดงสถานะของ p-code และสแต็กของ p-code ได้ด้วย

โอเวอร์เลย์ แมนเนเจอร์ : MOVE

ปัจจุบันจะเห็นได้ว่า โปรแกรมแต่ละโปรแกรมมีขนาดใหญ่ขึ้นเรื่อย ๆ เรื่องนี้ทำให้เกิดปัญหาบ้างเหมือนกัน วิธีแก้ปัญหาอย่างหนึ่งก็คือ ใช้โอเวอร์เลย์ (overlay) โปรแกรมที่ใช้เทคนิคนี้จะถูกแบ่งออกเป็นส่วน ๆ จะมีบางส่วนที่จำเป็นต้องใช้เท่านั้น ถูกโหลดเข้าไปยังเมมโมรี ส่วนไหนที่ยังไม่ได้ใช้จะไม่ถูกโหลดเข้ามา ซึ่งจะช่วยประหยัดเมมโมรีได้

C7.0 มีตัวจัดการโอเวอร์เลย์โดยมีชื่อว่า MOVE ซึ่งปรับปรุงไปมากเมื่อเทียบกับตัวจัดการโอเวอร์เลย์อื่น ๆ เช่น สนับสนุนพอยน์เตอร์ฟังก์ชัน, สนับสนุน XMS และ EMS caching เป็นต้น

ส่งท้าย

ถ้าจะเปรียบเทียบกับ Borland C++ 3.0 แล้ว ก็ต้องเปรียบคนละส่วนกัน เพราะทั้ง 2 คอมพายเลอร์มีข้อดีข้อเสียคนละด้าน Borland C++ 3.0 นั้นใช้ง่ายและเป็นโปรแกรมแบบ IDE (Turbo C++ for Windows) ส่วนของไมโครซอฟต์นั้นเป็นที่น่าแปลกใจว่าไมโครซอฟต์เป็นเจ้าของไมโครซอฟต์วินโดวส์แท้ ๆ แต่กลับมีโปรแกรมอย่าง PWB, codeView ที่รันบนดอส

MFC ของไมโครซอฟต์ค่อนข้างจะใช้ง่ายกว่า OWL ของบอร์แลนด์ นอกจากนั้นถ้าตัด OWL ออกไปแล้ว Borland C++ 3.0 จะมีราคาเท่ากัน (โดยประมาณ C7.0 ทั้งแพคเกจ)

Microsoft C/C++ 7.0 ได้นำไปสู่โลกของการเขียนโปรแกรมแนว objected-oriented ความสามารถของ C7.0 ที่เด่น ๆ ก็มี สนับสนุน C++, MFC, p-code, precompiled header เป็นต้น ในส่วนอื่น ๆ ก็เป็นไปตามสไตล์ไมโครซอฟต์คงต้องเป็นหน้าที่ของผู้อ่านล่ะครับว่า C7.0 มีความสามารถคุ้มเพียงใด